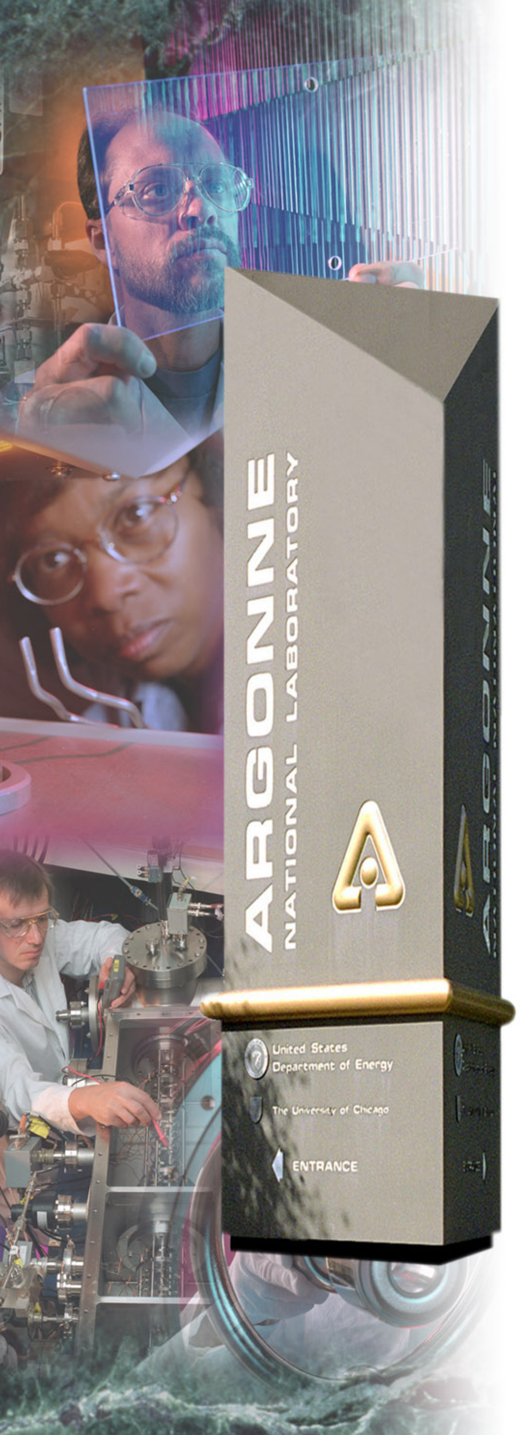


Using IDL and Python with EPICS

Mark Rivers, University of Chicago

Argonne National Laboratory

Office of Science
U.S. Department of Energy

A U.S. Department of Energy
Office of Science Laboratory
Operated by The University of Chicago

Outline

- Quick Overview of IDL
- ezca library
- Calling ezca from IDL
- IDL CA API
- IDL EPICS class libraries
- IDL applications
- Overview of Python
- Python class libraries
- Python applications
- Using EPICS from Visual Basic

Overview of IDL

- **A high-level interpreted *programming language* with vector and array primitives - sort of a cross between BASIC and APL**
- **Modern programming language**
  - Flow control
  - Data structures
  - Objects
- **All operators and most functions work on scalar, vector or array data of any data type.**
- **Data *visualization tool,* advanced built-in graphics**
  - 2-D plots
  - Contour plots
  - Surface plots
  - Shaded surfaces
  - Gray scale/color images
  - Isosurfaces
  - Volume rendering
- **Multi-platform support**
  - Unix: Sun, Hewlett Packard, Silicon Graphics, IBM
  - Linux
  - Microsoft Windows
  - Mac Darwin
- **List price: ~$3,000 on workstations, ~$1,500 on PC, Mac**

Overview of IDL

- **Can call external C or other code**
- **Very fast for array operations, as fast as compiled languages**
- **GUI builder**
- **Multi-threaded**
- **Good vendor support**
- **IDL GUI applications can be run at no cost (IDL 6.0 and above)**
  - Must have license to use IDL command line

Overview of IDL

Data Structures

- **A variable in IDL has both a structure and a data type associated with it. Both of these are dynamic, i.e. they can be changed via an assignment statement at any time.**
- **Data types**
  - Byte (b bit, unsigned)
  - Integer (16 bit, signed)
  - Long (32 bit, signed)
  - Float (32 bit floating point)
  - Double (64 bit floating point)
  - Complex (pair of 32 bit floats)
  - Double complex (pair of 64 bit floats)
  - String (0 to 64k characters)
- **Data Structures**
  - Scalar
  - Vector
  - Array - up to 7 dimensions
  - Structure - composed of other elements, like C
- **The sizes of arrays are limited only by the available virtual memory.**

Overview of IDL

Assignment Statements

```
A = B + 1
```

- **A has the same structure as B, with a data type equal to that of the most precise operand in the expression on the right hand side. In this case it could be any type except string.**
- **If B is a vector or array then 1 is added to each element.**

```
A = 0           ; A is a 16 bit integer
A = A * 0.5     ; A is now a 32 bit float
B = A(*,3)      ; B is equal to the 4th row of A
A(*,3) = 0      ; Set all elements in 4th row of A equal to 0
```

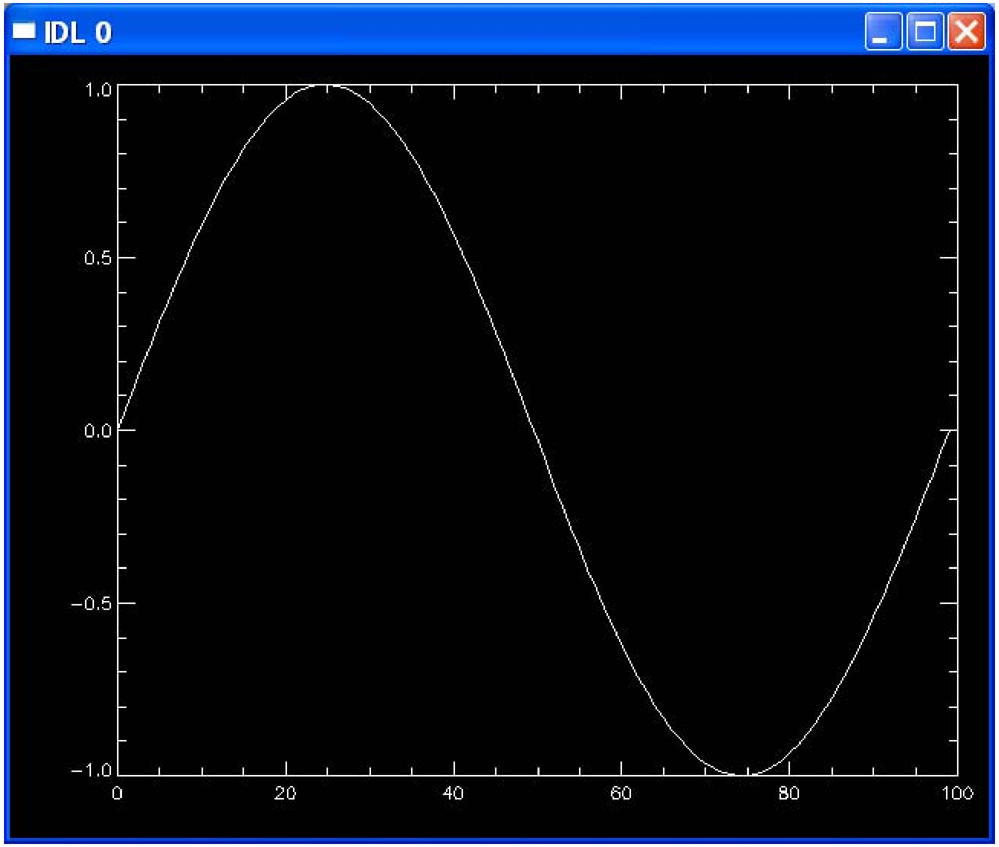
Syntax

- **Examples:**

```
image = fltarr(512, 512)          ; zero filled array
b = image(0:127, 0:127)           ; b is 128x128 array
image(*,100) = findgen(512)       ; replace row 100
plot, image(*,120)                ; plot row 121
; Display the power spectrum as an image
tvscl, alog(abs(fft(image, 1)))
```

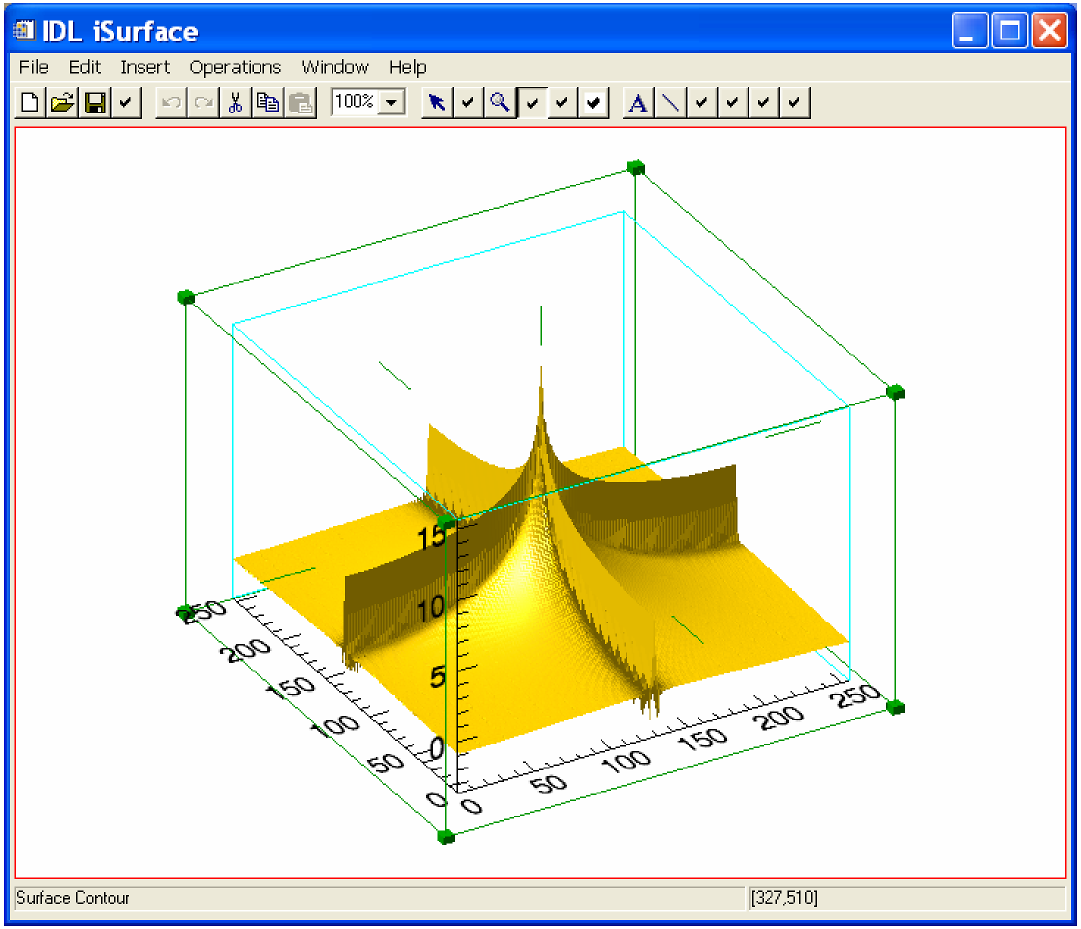
IDL Examples

```
IDL> a = sin(findgen(100)/99. * 2 * !pi)
IDL> help, a
A                FLOAT     = Array[100]
IDL> plot, a
```



IDL Examples

```
IDL> a = shift(alog(abs(fft(dist(256),1))),128,128)
IDL> isurface, a
```



ezca and EzcaScan

- **EPICS extensions for “Easy Channel Access”**
  - Don’t need to handle chids, just use PV name strings – hash table
  - Synchronous APIs - applications don’t have to handle callbacks
- **Ezca (partial list)**

```
-   epicsShareFunc int epicsShareAPI ezcaGet(char *pvname, char ezcatype,
-           int nelem, void *data_buff);
-   epicsShareFunc int epicsShareAPI ezcaPut(char *pvname, char ezcatype,
-           int nelem, void *data_buff);
-   epicsShareFunc int epicsShareAPI ezcaPutOldCa(char *pvname, char ezcatype,
-           int nelem, void *data_buff);
-   epicsShareFunc int epicsShareAPI ezcaNewMonitorValue(char *pvname,
-           char ezcatype); /* returns TRUE/FALSE */
-   epicsShareFunc int epicsShareAPI ezcaSetTimeout(float sec);
-   epicsShareFunc float epicsShareAPI ezcaGetTimeout(void);
-   epicsShareFunc int epicsShareAPI ezcaSetRetryCount(int retry);
-   epicsShareFunc int epicsShareAPI ezcaGetRetryCount(void);
-   epicsShareFunc int epicsShareAPI ezcaPvToChid(char *pvname, chid **cid);
-   epicsShareFunc int epicsShareAPI ezcaSetMonitor(char *pvname, char ezcatype);
-   epicsShareFunc int epicsShareAPI ezcaClearMonitor(char *pvname, char ezcatype);
-   epicsShareFunc int epicsShareAPI ezcaStartGroup(void);
-   epicsShareFunc int epicsShareAPI ezcaEndGroup(void);
-   epicsShareFunc int epicsShareAPI ezcaGetControlLimits(char *pvname,
-           double *low, double *high);
-   epicsShareFunc int epicsShareAPI ezcaGetGraphicLimits(char *pvname,
-           double *low, double *high);
-   epicsShareFunc int epicsShareAPI ezcaGetNelem(char *pvname, int *nelem);
-   epicsShareFunc int epicsShareAPI ezcaGetPrecision(char *pvname,
-           short *precision);
-   epicsShareFunc int epicsShareAPI ezcaGetStatus(char *pvname,
-           TS_STAMP *timestamp, short *status, short *severity);
-   epicsShareFunc int epicsShareAPI ezcaGetUnits(char *pvname,
-           char *units); /* units must be at least EZCA_UNITS_SIZE large */
```

ezca and EzcaScan

- **EzcaScan (partial list)**

  -
    ```
    epicsShareFunc int epicsShareAPI Ezca_getArray(int noName,char **pvName,int type,int nodata,void *value);
    ```

  -
    ```
    epicsShareFunc int epicsShareAPI Ezca_getArrayEvent(int noName,char **pvName,int type,int nodata,void *value);
    ```

  -
    ```
    epicsShareFunc int epicsShareAPI Ezca_putArray(int noName,char **pvName,int type ,int nodata,void *value);
    ```

  -
    ```
    epicsShareFunc int epicsShareAPI Ezca_putArrayEvent(int noName,char **pvName,int type,int nodata,void *value);
    ```

ezca and IDL

- **IDL can call “shareable libraries”, e.g. .so files on Unix, .dll files on Windows**
- **The argument passing convention is fixed, it is not compatible with ezca.dll directly**
- **Need a thin glue layer between IDL and ezca/EzcaScan**
- **ezcaIDL is the glue layer. Mostly just changes calling conventions. Provides a few functions that ezca and EzcaScan do not. Use the ezcaPvToChid() function.**
  - ezcaIDLGetEnumStrings()
  - ezcaIDLGetCountAndType()

IDL Channel Access API

Routines which return information about process variables

```
Status = caGet(pvname, value, /string, maximum=max)
Status = caGetControlLimits(pvname, low, high)
Status = caGetGraphicLimits(pvname, low, high)
Status = caGetPrecision(pvname, precision)
Status = caGetStatus(pvname, timestamp, status, severity)
Status = caGetUnits(pvname, units)
Status = caGetEnumStrings(pvname, strings)
Status = caGetCountAndType(pvname, count, type)
```

Routines which write new values to process variables

```
Status = caPut(pvname, value, wait=wait)
```

Routines which control channel access timeouts

```
Timeout = caGetTimeout()
caSetTimeout, timeout
RetryCount = caGetRetryCount()
caSetRetryCount, retrycount
```

IDL Channel Access API

Routines which control synchronous groups

```
caStartGroup
stat = caEndGroup(status)
```

Routines which control channel access monitors

```
Status = caSetMonitor(pvname)
Status = caClearMonitor(pvname)
State = caCheckMonitor(pvname)
```

Routines which control debugging and error messages

```
caDebug, state
caTrace, state
caError, err_string, /ON, /OFF, /PRINT, prefix=prefix
```

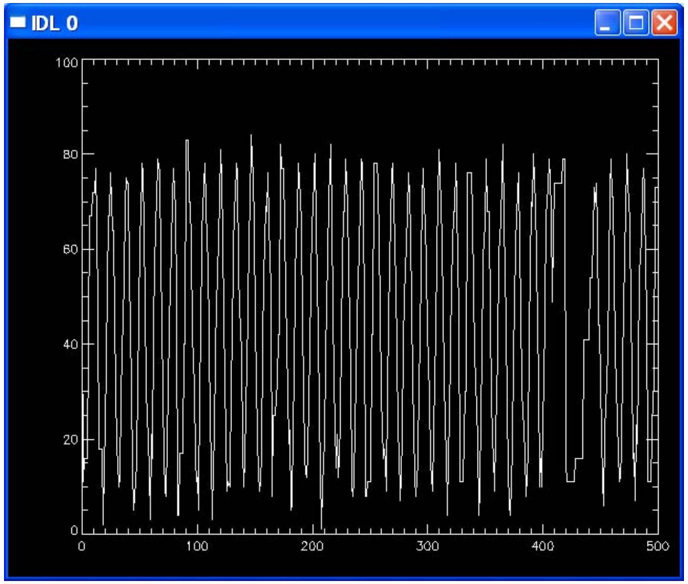
Documentation at

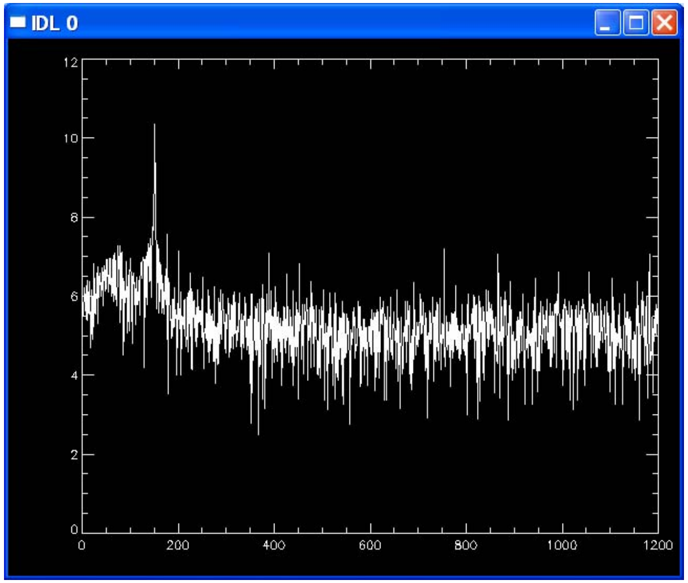
http://cars.uchicago.edu/software/idl/ezcaIDLGuide.html

http://cars.uchicago.edu/software/idl/ezcaIDLRef.html

IDL EPICS Examples

```
IDL> status = caget('13LAB:m1.VAL', position)
IDL> help, status, position
STATUS          LONG      =            0
POSITION        DOUBLE    =        517.19305
IDL> status = caget('13LAB:quadEM:mca1', spectrum)
IDL> plot, spectrum
IDL> help, status, spectrum
STATUS          LONG      =            0
SPECTRUM        LONG      = Array[2048]
IDL> plot, spectrum[0:500]
IDL> fft_data = alog(abs(fft(spectrum,1)))
IDL> plot, fft_data, xrange=[0,1023]
```





IDL EPICS Examples

Move a motor

```
DL> status = caput('13LAB:m8.VAL', 10000)
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
       215.52734
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
       835.64453
IDL> status = caget('13LAB:m8.RBV', pos)
IDL> print, pos
       1795.6055
```

Ezca timeout values are important!

```
IDL> print, cagettimeout()
    0.0200000
IDL> t0=systime(1)&for i=1,100 do t=caget('13LAB:m1', v)&print, systime(1)-t0
       2.9898720
IDL> casettimeout, .001
IDL> t0=systime(1)&for i=1,100 do t=caget('13LAB:m1', v)&print, systime(1)-t0
      0.21649790
```

IDL EPICS Examples

Using monitors

Monitored channels read the cached values on caget()

Can check whether a monitor has happened (a Channel Access value callback)

```
IDL> status = caSetMonitor('13LAB:m8.DMOV')
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =            1
IDL> status = caget('13LAB:m8.DMOV', done)
IDL> help, done
DONE            INT       =        1
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =            0
IDL> status = caput('13LAB:m8.VAL', 0)
IDL> state = caCheckMonitor('13LAB:m8.DMOV')
IDL> help, state
STATE           LONG      =            1
IDL> status = caget('13LAB:m8.DMOV', done)
IDL> help, state
STATE           LONG      =            1
IDL> help, done
DONE            INT       =        0
```

Monitors are useful for seeing that a PV changed state, even if its value is the same because one “missed” the transition. For example, PV goes 0->1->0. IDL polling might miss the one state, but checking a monitor would let one know that it happened.

IDL EPICS Class Libraries

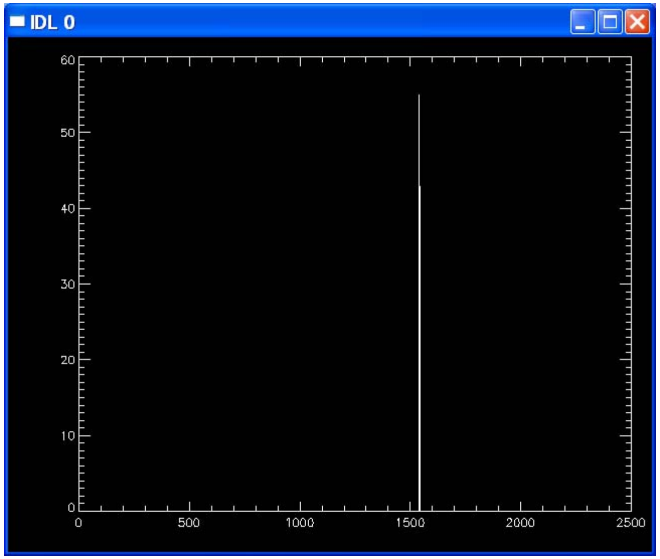
- **IDL object classes that hide the underlying EPICS process variables**
- **IDL objects treat all data as private, only accessible through methods.**
- **Provide an object-oriented interface to common beamline objects (motors, scalers, mcas, scans)**
  - epics_motor
  - epics_scaler
  - epics_mca (inherits device-independent mca class)
  - epics_med (multi-element detector)
  - epics_sscan
- **Example of epics_motor**

```
IDL> motor = obj_new('EPICS_MOTOR', '13LAB:m8')
IDL> motor->move, 1000.   ; Move to absolute position 10.
IDL> motor->wait         ; Wait for it to get there
```

IDL EPICS Class Libraries

- **Example of epics_mca**

```
IDL> mca = obj_new('epics_mca', '13LAB:aim_adc1')
IDL> mca->erase
IDL> mca->acquire_on
IDL> data = mca->get_data()
IDL> plot, data
```



- **Example of epics_scaler**

```
IDL> scaler = obj_new('epics_scaler', '13LAB:scaler1')
IDL> scaler->start, 10.    ; Count for 10 seconds
IDL> scaler->wait          ; Wait for it to get done
IDL> counts = scaler->read(); Read the counts on each channel
IDL> print, counts
    100000000    0      0      0      0      0      0      0
```

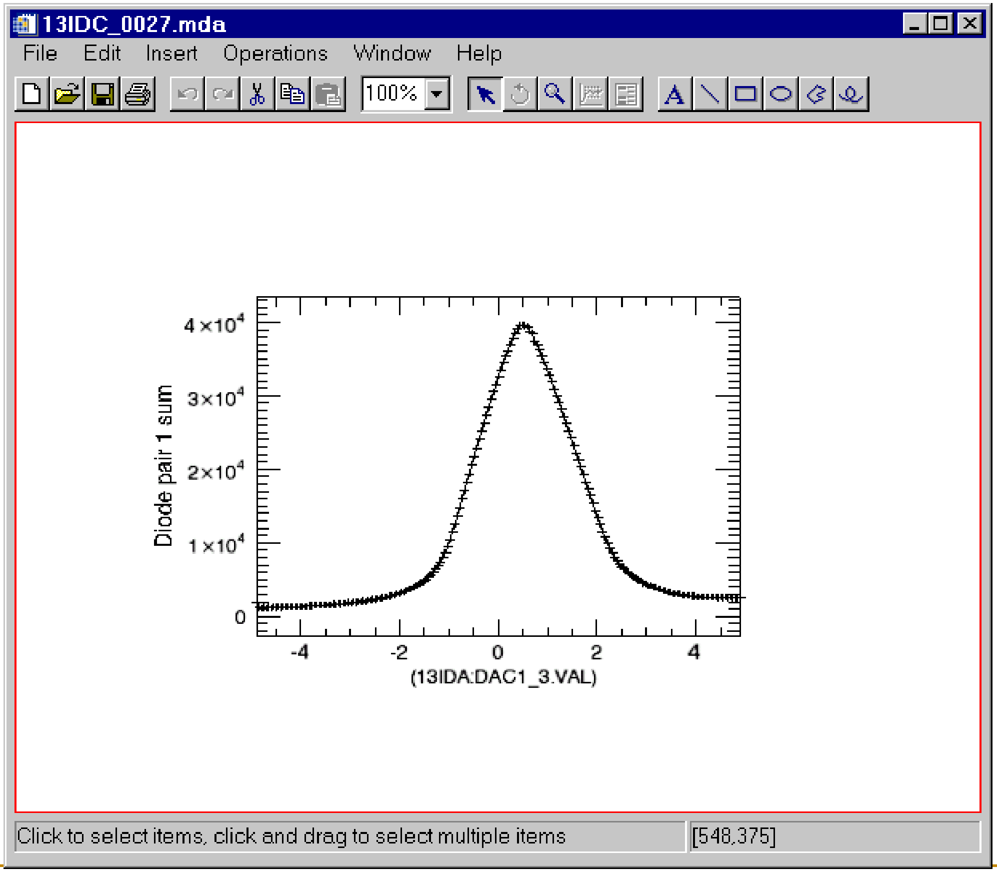
IDL EPICS Class Libraries

- **epics_sscan**
- **Designed to do the following:**
  - Provide an object-oriented interface to standard EPICS scans, enabling user written software to easily access scan header information and data.
  - Provide an easy way to read MDA files written by the saveData function in synApps.
  - Provide an easy way to get scan data into the IDL iTools system. iTools provide powerful interfaces for visualizing data, zooming in, adding annotation, and producing publication quality plots.
  - Provide a way to convert binary scan files (e.g. MDA) into ASCII
  - Does not currently communicate with the IOC for real-time data, but this is planned for the future

IDL EPICS Class Libraries

Example: Simple 1D epics_sscan

```
IDL> s = read_mda('13IDC_0027.mda') ; Read the data
IDL> s->display   ; Display the first detector
```

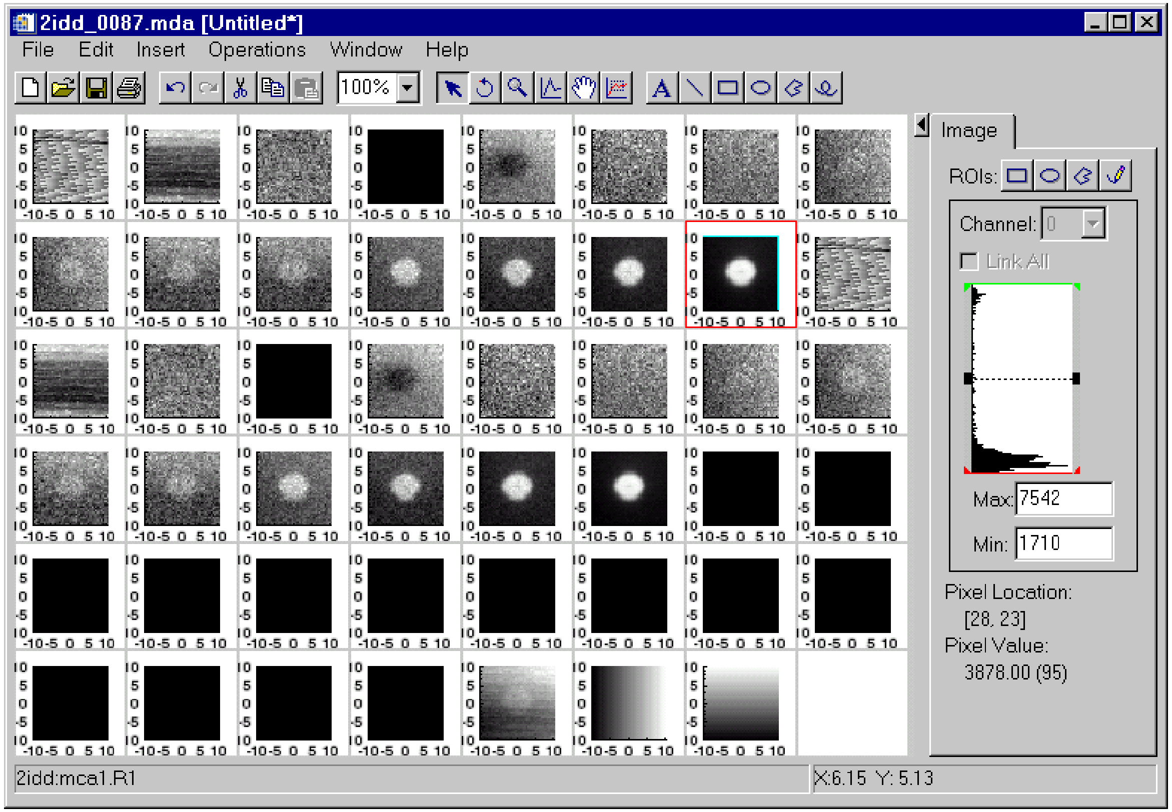


Click to select items, click and drag to select multiple items

IDL EPICS Class Libraries

Example: 2-D epics_sscan

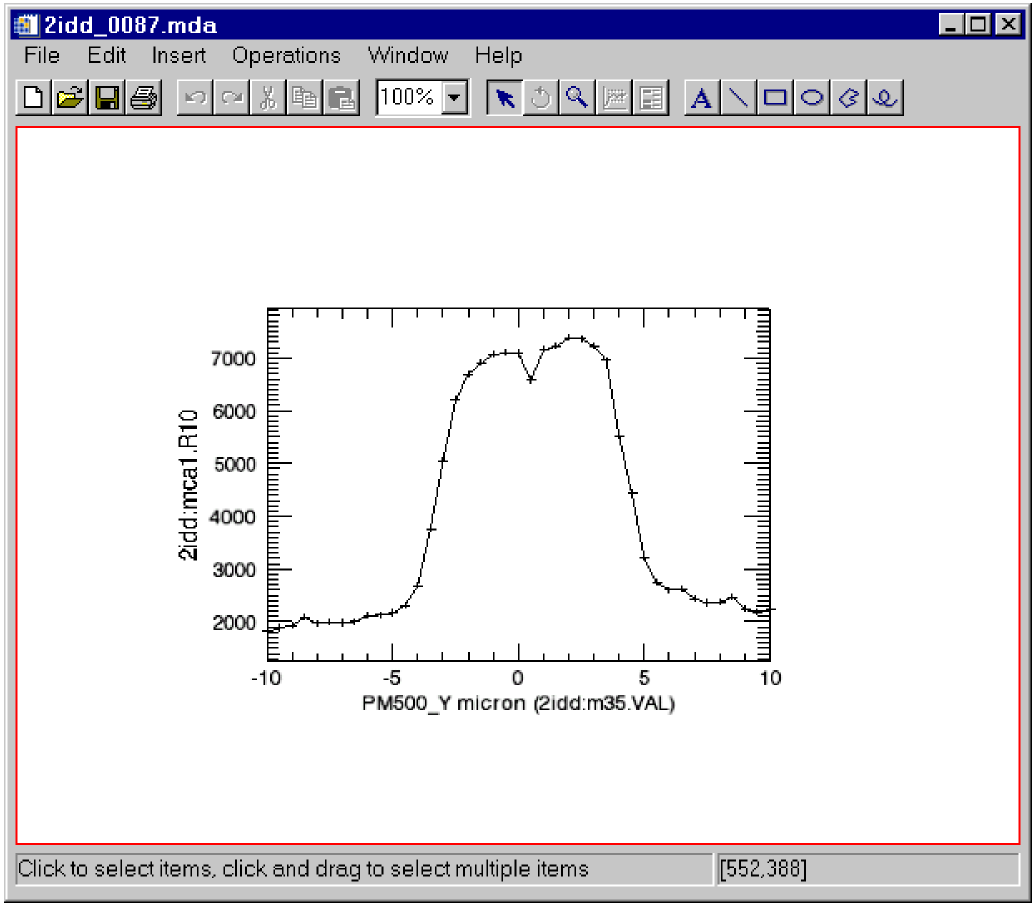
```
IDL> s=read_mda('2idd_0087.mda') ; Read the 2-D dataset
IDL> s->display, /all, /grid ; Display all of the
  images in a grid
```



IDL EPICS Class Libraries

Example: 2-D epics_sscan

```
; Plot a profile of column 20 (X=20) in detector 15.
IDL> s->display, detector=15, xrange=20
```



Click to select items, click and drag to select multiple items

[552,388]

IDL EPICS Class Libraries

Documentation: Reference manual for each class library

EPICS_MCA Class

This page was created by the IDL library routine mk_html_help. For more information on this routine, refer to the IDL Online Help Navigator or type:

? mk_html_help at the IDL command line prompt.

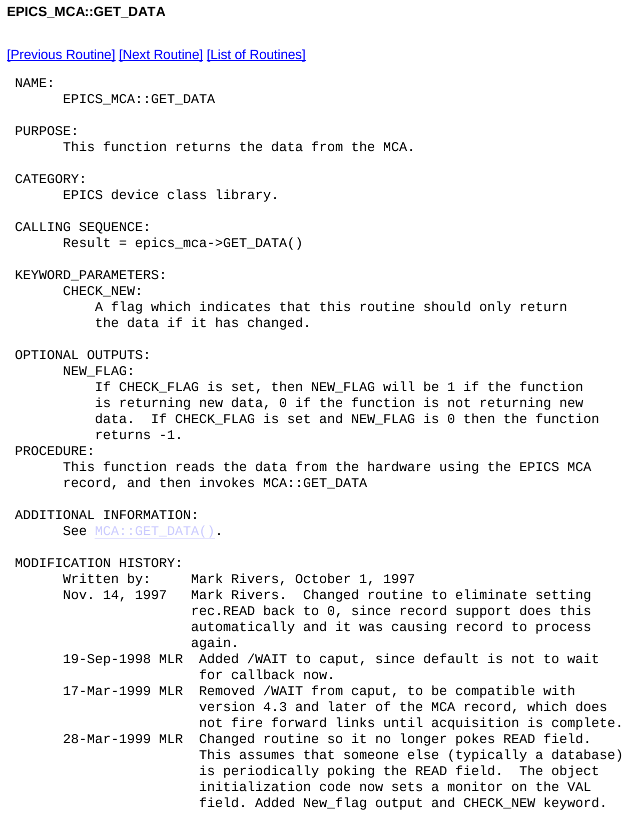
Last modified: Sat Jul 14 10:16:05 2001.

List of Routines

- EPICS_MCA::ACQUIRE_OFF
- EPICS_MCA::ACQUIRE_ON
- EPICS_MCA::ACQUIRE_WAIT
- EPICS_MCA::ADD_ROI
- EPICS_MCA::DEL_ROI
- EPICS_MCA::ERASE
- EPICS_MCA::GET_ACQUIRE_STATUS
- EPICS_MCA::GET_CALIBRATION
- EPICS_MCA::GET_DATA
- EPICS_MCA::GET_ELAPSED
- EPICS_MCA::GET_PRESETS
- EPICS_MCA::GET_ROIS
- EPICS_MCA::GET_ROI_COUNTS
- EPICS_MCA::GET_SEQUENCE
- EPICS_MCA::INIT
- EPICS_MCA::SET_CALIBRATION
- EPICS_MCA::SET_DATA
- EPICS_MCA::SET_PRESETS
- EPICS_MCA::SET_ROIS
- EPICS_MCA::SET_SEQUENCE
- EPICS_MCA::SPECTRA_SCAN
- EPICS_MCA::WRITE_FILE
- EPICS_MCA__DEFINE
- RELEASE_NOTES

Documentation at:

http://cars9.uchicago.edu/software/idl/



EPICS_MCA::GET_DATA

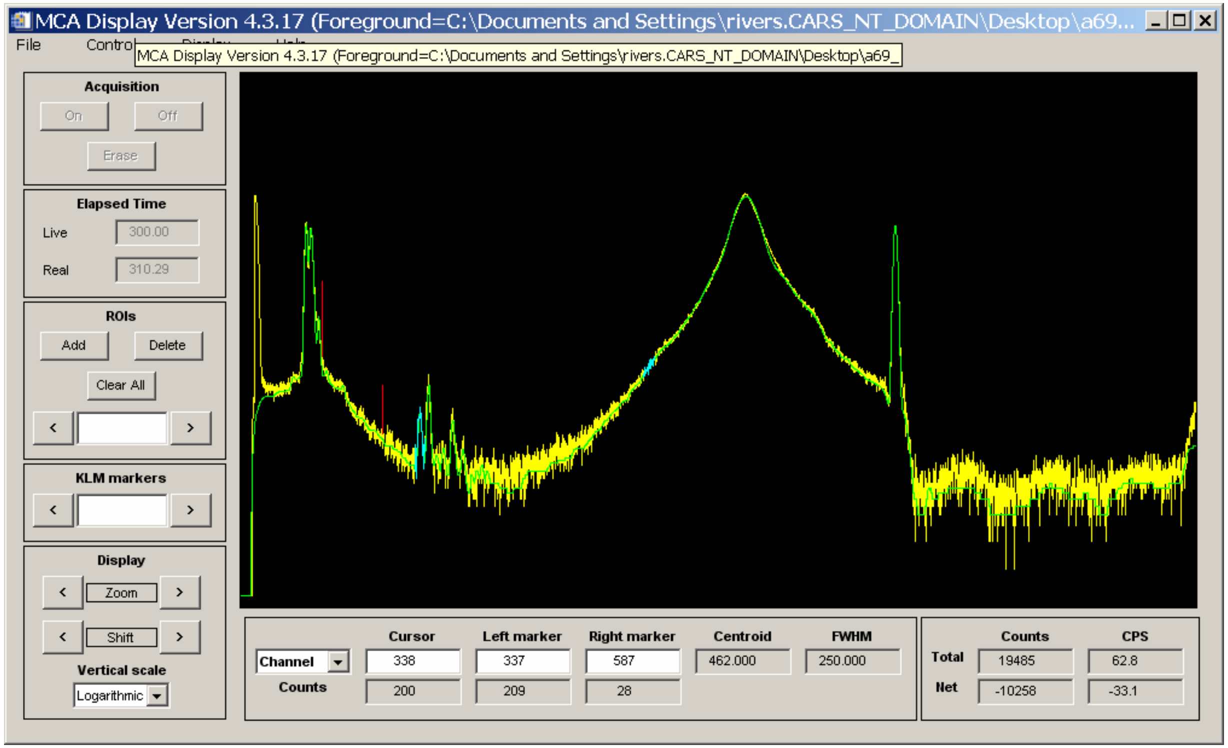
[Previous Routine] [Next Routine] [List of Routines]

```
NAME:
      EPICS_MCA::GET_DATA

PURPOSE:
      This function returns the data from the MCA.

CATEGORY:
      EPICS device class library.

CALLING SEQUENCE:
      Result = epics_mca->GET_DATA()

KEYWORD_PARAMETERS:
      CHECK_NEW:
          A flag which indicates that this routine should only return
          the data if it has changed.

OPTIONAL OUTPUTS:
      NEW_FLAG:
          If CHECK_FLAG is set, then NEW_FLAG will be 1 if the function
          is returning new data, 0 if the function is not returning new
          data.  If CHECK_FLAG is set and NEW_FLAG is 0 then the function
          returns -1.
PROCEDURE:
      This function reads the data from the hardware using the EPICS MCA
      record, and then invokes MCA::GET_DATA

ADDITIONAL INFORMATION:
      See MCA::GET_DATA().

MODIFICATION HISTORY:
      Written by:     Mark Rivers, October 1, 1997
      Nov. 14, 1997   Mark Rivers.  Changed routine to eliminate setting
                      rec.READ back to 0, since record support does this
                      automatically and it was causing record to process
                      again.
      19-Sep-1998 MLR  Added /WAIT to caput, since default is not to wait
                       for callback now.
      17-Mar-1999 MLR  Removed /WAIT from caput, to be compatible with
                       version 4.3 and later of the MCA record, which does
                       not fire forward links until acquisition is complete.
      28-Mar-1999 MLR  Changed routine so it no longer pokes READ field.
                       This assumes that someone else (typically a database)
                       is periodically poking the READ field.  The object
                       initialization code now sets a monitor on the VAL
                       field. Added New_flag output and CHECK_NEW keyword.
```

(See epics_mca__define.pro)

IDL EPICS Applications

- **mcaDisplay**
  - Full-featured program for displaying, controlling EPICS multi-channel analysers, including peak fitting
  - Uses epics_mca class library, and exports mca_display class, so it can be controlled by other IDL applications



MCA Display Version 4.3.17 (Foreground=C:\Documents and Settings\rivers.CARS_NT_DOMAIN\Desktop\a69...

File Contro

MCA Display Version 4.3.17 (Foreground=C:\Documents and Settings\rivers.CARS_NT_DOMAIN\Desktop\a69_

Acquisition

On Off

Erase

Elapsed Time

Live 300.00

Real 310.29

ROIs

Add Delete

Clear All

KLM markers

Display

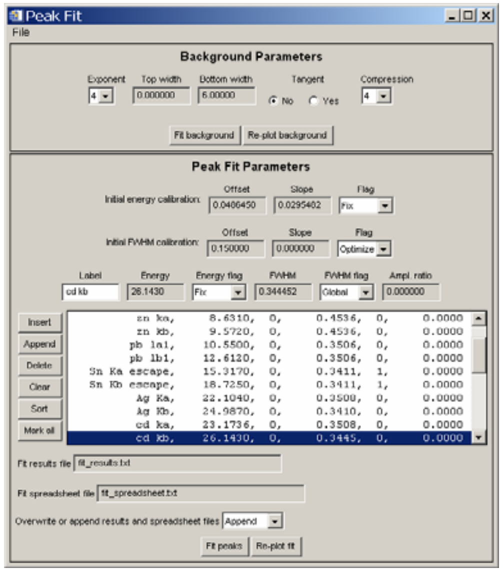
Zoom

Shift

Vertical scale Logarithmic

	Cursor	Left marker	Right marker	Centroid	FWHM
Channel	338	337	587	462.000	250.000
Counts	200	209	28		

	Counts	CPS
Total	19485	62.8
Net	-10258	-33.1



Peak Fit

File

Background Parameters

Exponent	Top width	Bottom width	Tangent	Compression
4	0.000000	6.00000	● No ○ Yes	4

Fit background Re-plot background

Peak Fit Parameters

	Offset	Slope	Flag
Initial energy calibration:	0.0486450	0.0295402	Fix
Initial FWHM calibration:	0.150000	0.000000	Optimize

Label	Energy	Energy flag	FWHM	FWHM flag	Ampl. ratio
cd kb	26.1430	Fix	0.344452	Global	0.000000

Insert Append Delete Clear Sort Mark all

```
          zn ka,     8.6310,  0,     0.4536,  0,     0.0000
          zn kb,     9.5720,  0,     0.4536,  0,     0.0000
         pb la1,    10.5500,  0,     0.3506,  0,     0.0000
         pb lb1,    12.6120,  0,     0.3506,  0,     0.0000
   Sn Ka escape,    15.3170,  0,     0.3411,  1,     0.0000
   Sn Kb escape,    18.7250,  0,     0.3411,  1,     0.0000
          Ag Ka,    22.1040,  0,     0.3500,  0,     0.0000
          Ag Kb,    24.9870,  0,     0.3410,  0,     0.0000
          cd ka,    23.1736,  0,     0.3508,  0,     0.0000
          cd kb,    26.1430,  0,     0.3445,  0,     0.0000
```

Fit results file fit_results.txt

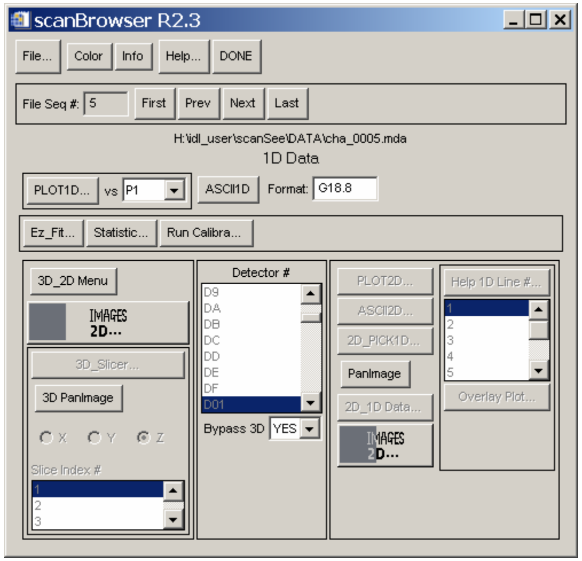
Fit spreadsheet file fit_spreadsheet.txt

Overwrite or append results and spreadsheet files Append

Fit peaks Re-plot fit

IDL EPICS Applications

Data catcher and data viewer (Ben-Chin Cha)


scanBrowser R2.3

File Seq #: 5

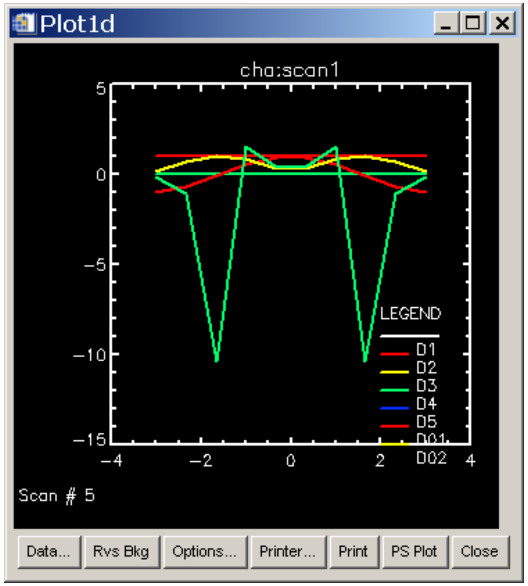
H:\idl_user\scanSee\DATA\cha_0005.mda

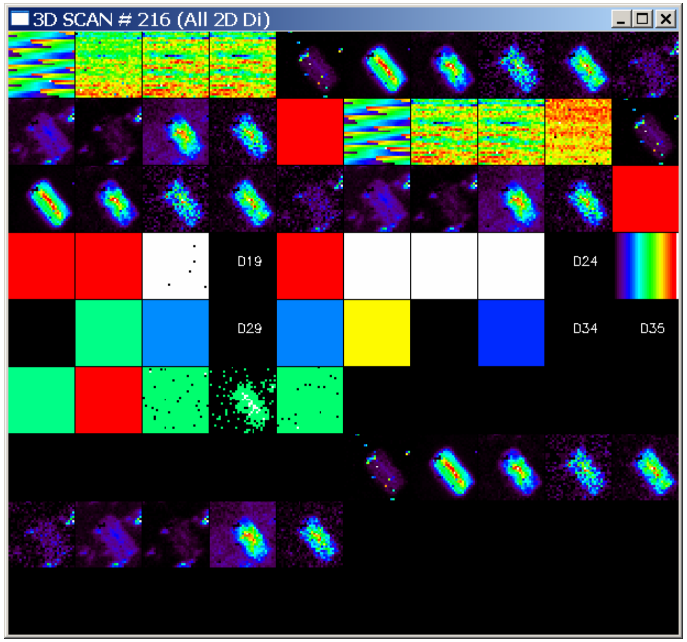
1D Data

PLOT1D... vs P1

Format: G18.8

Bypass 3D YES





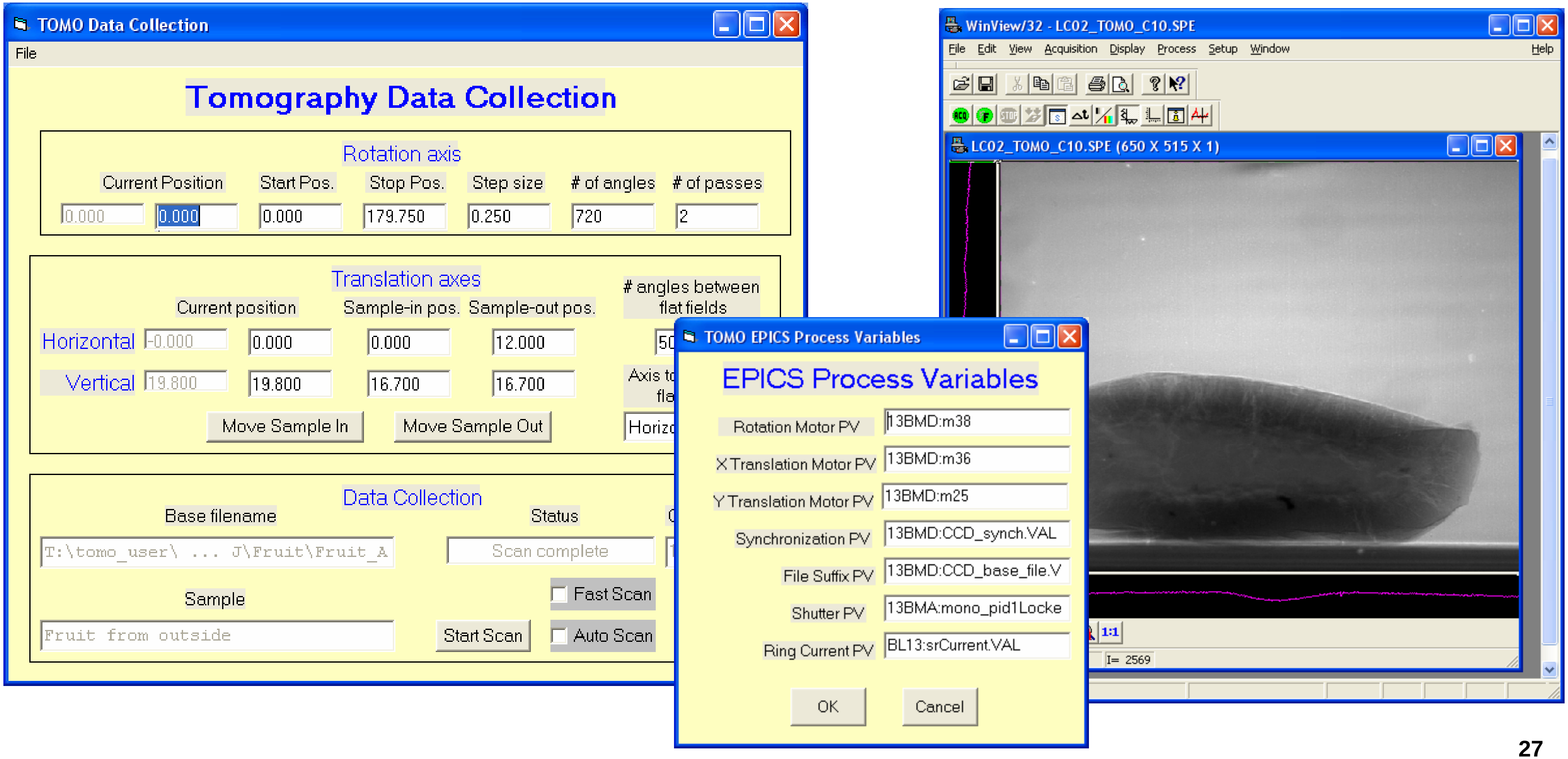
Using EPICS from Visual Basic

- **ezca.dll can be called directly from Visual Basic on Windows**
- **ezca.bas provides the interface**

```
Public Const ezcaByte   As Byte = 0
Public Const ezcaString As Byte = 1
Public Const ezcaShort  As Byte = 2
Public Const ezcaLong   As Byte = 3
Public Const ezcaFloat  As Byte = 4
Public Const ezcaDouble As Byte = 5

Public Declare Function ezcaGet Lib "ezca.dll" _
   (ByVal pvname As String, _
    ByVal ezcatype As Byte, _
    ByVal nelem As Long, _
    ByRef data As Any) As Long


Public Declare Function ezcaPut Lib "ezca.dll" Alias "ezcaPutOldCa" _
   (ByVal pvname As String, _
    ByVal ezcatype As Byte, _
    ByVal nelem As Long, _
    ByRef data As Any) As Long

Public Declare Function ezcaPutString Lib "ezca.dll" Alias "ezcaPutOldCa" _
   (ByVal pvname As String, _
    ByVal ezcatype As Byte, _
    ByVal nelem As Long, _
    ByVal data As Any) As Long

Public Declare Function ezcaPutCallback Lib "ezca.dll" Alias "ezcaPut" _
   (ByVal pvname As String, _
    ByVal ezcatype As Byte, _
    ByVal nelem As Long, _
    ByRef data As Any) As Long
```

Using EPICS from Visual Basic

- **Example: tomography data collection. VB used because it can easily control Roper’s WinView program for the CCD detector**



TOMO Data Collection

File

Tomography Data Collection

Rotation axis

Current Position		Start Pos.	Stop Pos.	Step size	# of angles	# of passes
0.000	0.000	0.000	179.750	0.250	720	2

Translation axes

	Current position		Sample-in pos.	Sample-out pos.
Horizontal	-0.000	0.000	0.000	12.000
Vertical	19.800	19.800	16.700	16.700

angles between flat fields: 50

Axis to ... fla...: Horiz...

Move Sample In · Move Sample Out

Data Collection

Base filename: T:\tomo_user\ ... J\Fruit\Fruit_A

Status: Scan complete

Sample: Fruit from outside

Start Scan · □ Fast Scan · □ Auto Scan

TOMO EPICS Process Variables

EPICS Process Variables

Rotation Motor PV	13BMD:m38
X Translation Motor PV	13BMD:m36
Y Translation Motor PV	13BMD:m25
Synchronization PV	13BMD:CCD_synch.VAL
File Suffix PV	13BMD:CCD_base_file.V
Shutter PV	13BMA:mono_pid1Locke
Ring Current PV	BL13:srCurrent.VAL

OK · Cancel

WinView/32 - LC02_TOMO_C10.SPE

LC02_TOMO_C10.SPE (650 X 515 X 1)

I= 2569